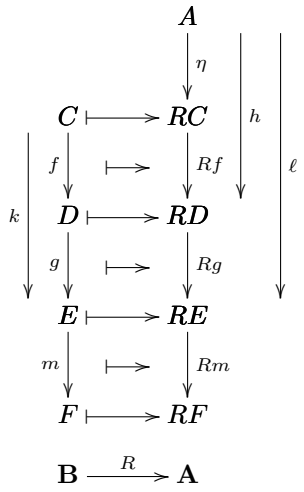
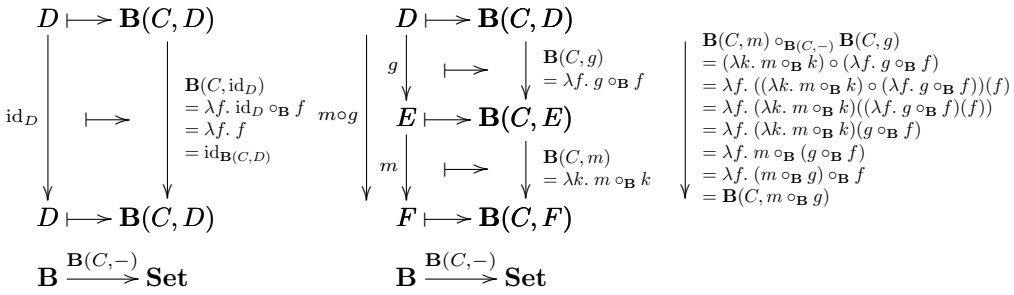
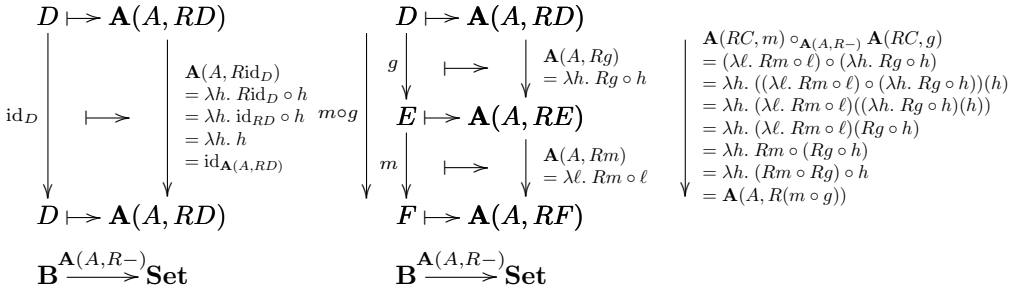

A
η
C ⟼ RC
h
f ⟼ Rf
ℓ
k
D ⟼ RD
g ⟼ Rg
E ⟼ RE
m ⟼ Rm
F ⟼ RF
B —R→ A


D ⟼ B(C, D)
id_D ⟼ B(C, id_D) = λf. id_D ∘_B f = λf. f = id_{B(C,D)}
D ⟼ B(C, D)
B —B(C,−)→ Set
D ⟼ B(C, D)
g ⟼ B(C, g) = λf. g ∘_B f
E ⟼ B(C, E)
m ⟼ B(C, m) = λk. m ∘_B k
F ⟼ B(C, F)
B —B(C,−)→ Set
m∘g
B(C, m) ∘_{B(C,−)} B(C, g)
= (λk. m ∘_B k) ∘ (λf. g ∘_B f)
= λf. ((λk. m ∘_B k) ∘ (λf. g ∘_B f))(f)
= λf. (λk. m ∘_B k)((λf. g ∘_B f)(f))
= λf. (λk. m ∘_B k)(g ∘_B f)
= λf. m ∘_B (g ∘_B f)
= λf. (m ∘_B g) ∘_B f
= B(C, m ∘_B g)


D ⟼ A(A, RD)
id_D ⟼ A(A, Rid_D) = λh. Rid_D ∘ h = λh. id_{RD} ∘ h = λh. h = id_{A(A,RD)}
D ⟼ A(A, RD)
B —A(A,R−)→ Set
D ⟼ A(A, RD)
g ⟼ A(A, Rg) = λh. Rg ∘ h
E ⟼ A(A, RE)
m ⟼ A(A, Rm) = λℓ. Rm ∘ ℓ
F ⟼ A(A, RF)
B —A(A,R−)→ Set
m∘g
A(RC, m) ∘_{A(A,R−)} A(RC, g)
= (λℓ. Rm ∘ ℓ) ∘ (λh. Rg ∘ h)
= λh. ((λℓ. Rm ∘ ℓ) ∘ (λh. Rg ∘ h))(h)
= λh. (λℓ. Rm ∘ ℓ)((λh. Rg ∘ h)(h))
= λh. (λℓ. Rm ∘ ℓ)(Rg ∘ h)
= λh. Rm ∘ (Rg ∘ h)
= λh. (Rm ∘ Rg) ∘ h
= A(A, R(m ∘ g))

```
module 2022Cats3 where

open import Level
open import Relation.Binary.PropositionalEquality as Eq
open Eq.≡-Reasoning

postulate ext : ∀ {ℓ ℓ'} {A : Set ℓ} {B : A → Set ℓ'} {f g : (a : A) → B a}
                  → (∀ (a : A) → f a ≡ g a) → f ≡ g

module Cats3 where

record Cat {ℓ₀ ℓ₁ : Level} : Set (suc (ℓ₀ ⊔ ℓ₁)) where
  field
    Objs  : Set ℓ₀
    Hom   : Objs → Objs → Set ℓ₁
    id    : (D : Objs) → (Hom D D)
    _▫_   : {C D E : Objs}
            → (g : Hom D E)
            → (f : Hom C D)
            →   (Hom C E)
    idL   : {D E : Objs}
            → (g : Hom D E)
            → (g ≡ g ▫ (id D))
    idR   : {D E : Objs}
            → (g : Hom D E)
            → (g ≡ (id E) ▫ g)
    assoc : {C D E F : Objs}
            → (f : Hom C D)
            → (g : Hom D E)
            → (m : Hom E F)
            → (m ▫ (g ▫ f) ≡ (m ▫ g) ▫ f)

catSet : Cat {suc zero} {zero}
Cat.Objs  catSet = Set
Cat.Hom   catSet = λ A B → (A → B)
Cat.id    catSet = λ A → (λ a → a)
Cat._▫_   catSet = λ g f → (λ a → g (f a))
Cat.idL   catSet = λ f → refl
Cat.idR   catSet = λ f → refl
Cat.assoc catSet = λ f g h → refl

catSetH : Cat {suc zero} {zero}
Cat.Objs  catSetH = Set
Cat.Hom   catSetH = λ A B → (A → B)
Cat.id    catSetH = λ A → (λ a → a)
Cat._▫_   catSetH = λ g f → (λ a → g (f a))
Cat.idL   catSetH = λ f → refl
```

```
Cat.idR    catSetH = λ f → refl
Cat.assoc catSetH = λ f g h → refl

record Functor {ℓ₀ ℓ₁ ℓ₂ ℓ₃ : Level}
               (catB : Cat {ℓ₀} {ℓ₁})
               (catA : Cat {ℓ₂} {ℓ₃})
             : Set (suc (ℓ₀ ⊔ ℓ₁ ⊔ ℓ₂ ⊔ ℓ₃)) where
  field
    ac0      : (Cat.Objs catB) → (Cat.Objs catA)
    ac1      : {C D : Cat.Objs catB}
               → (Cat.Hom catB C D)
               → (Cat.Hom catA (ac0 C) (ac0 D))
    respids  : {C : Cat.Objs catB}
               → (ac1 (Cat.id catB C) ≡ Cat.id catA (ac0 C))
    -- respcomp : {C D E : Cat.Objs catB}
    -- → {f : Cat.Hom catB C D}
    -- → {g : Cat.Hom catB D E}
    -- → (ac1 (Cat._∘_ catB g f) ≡ (Cat._∘_ catA (ac1 g) (ac1 f)))
  --
  -- R.ac0 C = RC
  -- R.ac1 {C} {D} f = R f
  -- R.respids {C} : R(id C) ≡ id(RC)
  -- R.respcomp {C} {D} {E} {f} {g} : R(g∘f) ≡ Rg∘Rf

record NT {ℓ₀ ℓ₁ ℓ₂ ℓ₃ : Level}
          {catC : Cat {ℓ₀} {ℓ₁}}
          {catD : Cat {ℓ₂} {ℓ₃}}
          (F G : Functor catC catD)
            : Set (suc (suc (ℓ₀ ⊔ ℓ₁ ⊔ ℓ₂ ⊔ ℓ₃))) where
  field
    ac       : (A : Cat.Objs catC)
               → (Cat.Hom catD (Functor.ac0 F A) (Functor.ac0 G A))
    sqcond   : {A B : Cat.Objs catC}
               → (f : Cat.Hom catC A B)
               → ((Cat._▫_ catD (ac B) (Functor.ac1 F f)) ≡
                    (Cat._▫_ catD (Functor.ac1 G f) (ac A)))
               → Set
  --
  -- T.ac A : Hom(FA,GA)
  -- T.sqcond {A} {B} f : TB∘Ff ≡ Gf∘TA

postulate catA : Cat {zero} {zero}
postulate catB : Cat {zero} {zero}
postulate R : Functor catB catA
postulate A : Cat.Objs catA
postulate C : Cat.Objs catB
postulate η : Cat.Hom catA A (Functor.ac0 R C)
```

```
postulate D E F : Cat.Objs catB
postulate f : Cat.Hom catB C D
postulate g : Cat.Hom catB D E

catB[C,-] : Functor catB catSetH
Functor.ac0     catB[C,-] D = Cat.Hom catB C D
Functor.ac1     catB[C,-] {D} {E} g f = Cat._▫_ catB g f
Functor.respids catB[C,-] {D} = trans (ext (λ f → sym (Cat.idR catB f))) refl
-- Functor.respcomp catB[C,-] {D} {E} {F} {g} {m} = {!!}

infix 10 _▫B_

B₀      : Set
B₀      = Cat.Objs catB
B[_,_] : B₀ → B₀ → Set
B[_,_] D E = Cat.Hom catB D E
_▫B_ : {D E F : B₀} → (B[_,_] E F) → (B[_,_] D E) → (B[_,_] D F)
_▫B_ {D} {E} {F} m g = Cat._▫_ catB m g
id_B   : (D : B₀) → B[_,_] D D
id_B   D = Cat.id catB D

Set[_,_] : Set → Set → Set
Set[_,_] A B = Cat.Hom catSet A B

B[C,_]₀ : B₀ → Set
B[C,_]₀ E = Functor.ac0 catB[C,-] E
B[C,_]₁ : {D E : B₀} → (B[_,_] D E) → (B[_,_] C D) → (B[_,_] C E)
B[C,_]₁ {D} {E} g = Functor.ac1 catB[C,-] g

module On_DEF (D E F : Cat.Objs catB) where
  idD           : B[_,_] D D
  idD           = id_B D
  B[C,D]        : Set
  B[C,D]        = B[C,_]₀ D
  B[C,idD]      : Set[_,_] B[C,D] B[C,D]
  B[C,idD]      = B[C,_]₁  idD
  id[B[C,D]]    : Set[_,_] B[C,D] B[C,D]
  id[B[C,D]]    = Cat.id  catSet B[C,D]
  λf⇒[idD▫f]  : (f : B[C,D]) → B[C,D]
  λf⇒[idD▫f]  f = idD ▫B f
  λf⇒f          : (f : B[C,D]) → B[C,D]
  λf⇒f          f = f
  λf⇒<idD▫f≡f> : (f : B[C,D]) → (idD ▫B f) ≡ f
  λf⇒<idD▫f≡f> f = sym (Cat.idR catB f)
  --
  <B[C,idD]≡[λf⇒[idD▫f]]> : B[C,idD] ≡ λf⇒[idD▫f]
```

```
  <B[C,idD]≡[λf⇒[idD ▫ f]]> = refl
  <[λf⇒[idD ▫ f]]≡[λf⇒f]> : λf⇒[idD ▫ f] ≡ λf⇒f
  <[λf⇒[idD ▫ f]]≡[λf⇒f]> = ext λf⇒<idD ▫ f≡f>
  <[λf⇒f]≡id[B[C,D]]>     : λf⇒f ≡ id[B[C,D]]
  <[λf⇒f]≡id[B[C,D]]>     = refl
  --
  <B[C,idD]≡id[B[C,D]]> : B[C,idD] ≡ id[B[C,D]]
  <B[C,idD]≡id[B[C,D]]> = begin
    B[C,idD]      ≡⟨ <B[C,idD]≡[λf⇒[idD ▫ f]]> ⟩
    λf⇒[idD ▫ f]  ≡⟨ <[λf⇒[idD ▫ f]]≡[λf⇒f]> ⟩
    λf⇒f          ≡⟨ <[λf⇒f]≡id[B[C,D]]>     ⟩
    id[B[C,D]]
    ■

-- (find-es "agda" "module-system")

module M = On_DEF D E F -- where
open M

-- foo : {!Set!}
-- foo = <B[C,idD]≡id[B[C,D]]>
-- trans (ext (λ f → sym (Cat.idR catB f))) refl


  --
  -- [λf⇒[id:D∘f]]≡[λf⇒f] : λf⇒[id:D∘f] ≡ λf⇒f
  -- [λf⇒[id:D∘f]]≡[λf⇒f] = {!!}
```